UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/827,971	04/06/2001	Jason Souloglou		5417

36183 7590 08/11/2004

PAUL, HASTINGS, JANOFSKY & WALKER LLP
P.O. BOX 919092
SAN DIEGO, CA 92191-9092

EXAMINER	
CHOW, CHIH CHING	
ART UNIT	**PAPER NUMBER**
2122	6

DATE MAILED: 08/11/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/827,971	SOULOGLOU ET AL.	
	Examiner	Art Unit	
	Chih-Ching Chow	2122	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *06 April 2001*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-18* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-18* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *06 April 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☒ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date *08/24/01*.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

Priority

1. It is noted that this application appears to claim subject matter disclosed in prior Application No. PCT/GB99/03168, filed 11 October 1999, and prior Application No. (GB) 9822075.9, filed 10 October 1998. A reference to the prior application must be inserted as the first sentence of the specification of this application or in an application data sheet (37 CFR 1.76), if applicant intends to rely on the filing date of the prior application under 35 U.S.C. 119(e) or 120. See 37 CFR 1.78(a). For benefit claims under 35 U.S.C. 120, the reference must include the relationship (i.e., continuation, divisional, or continuation-in-part) of all nonprovisional applications. Also, the current status of all nonprovisional parent applications referenced should be included.

2. Acknowledgment is made of applicant's claim for foreign priority based on an application filed in Application No. PCT/GB99/03168 on 11 October 1999 and a prior Application No. (GB) 9822075.9, filed 10 October 1998. It is noted, however, that applicant has not filed a certified copy of either the Application No. PCT/GB99/03168 application nor the Application No. (GB) 9822075.9 application as required by 35 U.S.C. 119(b).

3. Since this application is a CIP (Continuation-In-Part), for priority purpose, the priority date is 06 April, 2001.

Drawings

4. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(4) because reference characters 1, 2 (FIG 1), 3 have been used to designate FIGs. 1-6 (different parts with same number). Corrected drawing sheets are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

5. The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they include the following reference character(s) not mentioned in the description: characters 1-12 in FIGs 1-5. The so-called 'expression forest' (page 15, 8th paragraph of the specification) is not labeled in any of the figures 1 to 5. Corrected drawing sheets, or amendment to the specification to add the reference character(s) in the description, are required in reply to the Office action to avoid abandonment of the application. Any amended replacement drawing sheet should include all of the figures appearing on the immediate prior version of the sheet, even if only one figure is being

amended. The replacement sheet(s) should be labeled "Replacement Sheet" in the page header (as per 37 CFR 1.84(c)) so as not to obstruct any portion of the drawing figures. If the changes are not accepted by the examiner, the applicant will be notified and informed of any required corrective action in the next Office action. The objection to the drawings will not be held in abeyance.

Specification

6. The following guidelines illustrate the preferred layout for the specification of a utility application. These guidelines are suggested for the applicant's use.

Arrangement of the Specification

As provided in 37 CFR 1.77(b), the specification of a utility application should include the following sections in order. Each of the lettered items should appear in upper case, without underlining or bold type, as a section heading. If no text follows the section heading, the phrase "Not Applicable" should follow the section heading:

(a) TITLE OF THE INVENTION.

(b) CROSS-REFERENCE TO RELATED APPLICATIONS.

(c) STATEMENT REGARDING FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT.

(d) INCORPORATION-BY-REFERENCE OF MATERIAL SUBMITTED ON A COMPACT DISC (See 37 CFR 1.52(e)(5) and MPEP 608.05. Computer program listings (37 CFR 1.96(c)), "Sequence Listings" (37 CFR 1.821(c)), and tables having more than 50 pages of text are permitted to be submitted on compact discs.) or

REFERENCE TO A "MICROFICHE APPENDIX" (See MPEP § 608.05(a). "Microfiche Appendices" were accepted by the Office until March 1, 2001.)

(e) BACKGROUND OF THE INVENTION.

(1) Field of the Invention.

(2) Description of Related Art including information disclosed under 37 CFR 1.97 and 1.98.

(f) BRIEF SUMMARY OF THE INVENTION.

(g) BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S).

(h) DETAILED DESCRIPTION OF THE INVENTION.

(i) CLAIM OR CLAIMS (commencing on a separate sheet).

(j) ABSTRACT OF THE DISCLOSURE (commencing on a separate sheet).

(k) SEQUENCE LISTING (See MPEP § 2424 and 37 CFR 1.821-1.825. A "Sequence Listing" is required on paper if the application discloses a nucleotide or amino acid sequence as defined in 37 CFR 1.821(a) and if the required "Sequence Listing" is not submitted as an electronic document on compact disc).

Double Patenting

7. A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process ... may obtain a patent therefore ..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957); and *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the conflicting claims so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

8. Claims 1-18 provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-18 of copending Application No. 10/164,772. This is a provisional double patenting rejection since the conflicting claims have not in fact been patented.

9. Claims 1-18 provisionally rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-18 of copending Application No. 10/165,378. This is a

provisional double patenting rejection since the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 102

10. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

11. Claims 1-3, 5-7, 12, 14, 16-18 are rejected under 35 U.S.C. 102(b) as being anticipated by Aho et al, "Compiler, principles, techniques, and tools" book, published in 1986 (herein after Aho).

CLAIMS	Aho
1. A method of generating an intermediate representation of program code, the method comprising the computer implemented steps of: a. generating a plurality register objects representing abstract registers, a single register object representing a respective abstract register; and b. generating expression objects each representing a different element of said program code as that element arises in the program code, each expression object being referenced by a register object which it relates either directly, indirectly via references from other expression objects.	With respect to Claim 1, first paragraph, Aho discloses the concept of generating **intermediate representation** of program code in his book, on page 12, section 'Intermediate Code Generation': "After syntax and semantic analysis, some computers generate an explicit **intermediate representation** of the **source program**. We can think of this **intermediate representation** as a program for an abstract machine." With respect to paragraphs a and b. a. Registers are inherited from any microprocessors, used to hold data for a particular purpose. On Aho's book, Aho further discloses the use of registers for any element of program code. In order for Aho to use those registers, those registers must be **generated** at some point. On page 517, section 'Register Allocation': "efficient utilization of registers is

particularly important in generating good code. The use of registers is often subdivided into two subprograms:

1. During register allocation, we select the set of variables that will reside in registers at the point in the program.

2. During a subsequent register assignment phase, we pick the specific register that a variable will reside in. "

Aho's variables are held by **register objects,** they also imply to 'abstract registers' since they can hold any type of data (abstract data types) ;

b. an **expression object** is an operation performed for the register objects. The **expression objects** should be generated when the parsing of the program code. On Aho's book, page 49, under 'Abstract and Concrete Syntax' section, "A useful starting point for thinking about the translation of an input string is an *abstract syntax tree* in which each node represents

an operator and the children of the node represent the operands." Here the node can be considered as a **register object** representing a respective **abstract register** and the operator is an **expression object** that is referenced by a **register object** (operand). An example is given in Aho's book, page 558-559, each of the t2, t3, t1 and t4 are 'expression objects' and they are either relates directly, or indirectly via references from other expression objects. (E.g. t1-4 are all **expression objects**, where t2 is indirectly related to t4).

2. A method according to claim 1 wherein said program code is expressed in terms of an instruction set of a subject processor.

For the feature of claims 1 see Aho. Aho's book, page 14, "we consider an intermediate form called 'three-address code,' which is like the assembly language for a machine in which every memory location can act like a register. Three-address code consists of **a sequence of**

	instructions, each of which has at most three operands." This paragraph implies that the original program code is expressed in terms of an **instruction set** of a **subject processor** and a translation will be done for a target processor.
3. A method according to claim 2, wherein said register objects represent abstract registers corresponding to registers of said subject processor.	For the feature of claim 1 and 2 see claim 1 and 2 rejections; as to the "abstract registers corresponding to registers of said subject processor" part, it is implied under claim 2 rejection, "which is like the assembly language for a machine in which every memory location can act like a register. " – Where all the original registers (memory location) are corresponding to the subject processor.
5. A method according to claim 1, wherein at least some said expression objects feed into more than one said register object.	For the feature of claim 1 see Aho. Again, see Aho, page 559, Fig. 9.20, each of the t2, t3, t1 and t4 are 'expression objects' and they are feed into more than one

register object. (E.g. t1 is an expression object, it feed into register objects a and b; t2, is also an expression object, it feed into register objects c and d). Any program would have some expression objects since the program should perform certain functions, functions are 'operations' and they are represented by 'expression objects'; operands feed into operations, here operands are represented by 'register objects'.

6. A method according to claim 1, wherein said expression objects are not duplicated.

For the feature of claim 1 see Aho. On Aho page 290, under 'Directed Acyclic Graphs for Expressions' section, "A directed acyclic graph (hereafter called a dag) for an expression identifies the common subexpressions in the expression. Like a syntax tree, a dag has a node for every subexpression of the expression; an interior node represents an operator and its children represent its operands." In

addition, on page 291, first paragraph, "A dag is obtained if the function constructing a node first **checks to see whether an identical node already exists**. ... if so, mknode can return a pointer to the previously constructed node" – the duplication of a function/operation node (expression object) is checked before a new function node is created.

7. A method according to claim 1, wherein a single said expression object is generated for a given element of said program code, and each said expression object is referenced by said register objects to which relates.

For the feature of claim 1 see Aho. As mentioned previously, any element of a program code can be represented by an expression object (a function/operation needs to be done). See rejection on claim 1 for the rest of the claim.

12. The method of claim 1, comprising translating the program code written for execution by a processor of a first type so that the program code may be executed by a processor of a second type, using the

For the feature of claim 1 see Aho. On Aho, page 463, first paragraph, "In the analysis-synthesis model of a compiler, the front end translates a source program into an **intermediate representation**

generated intermediate representation.

form which the back end generates target code. Although a source program can be translated directly into the target language, some benefits of using a machine-independent intermediate form are:

1. Retargeting is facilitated; a compiler for a different machine can be created by attaching a back end of the new machine to an existing front end.

2. A machine-independent code optimizer can be done to the intermediate representation." Aho taught us that the program code from a front end processor can be translated into **intermediate representation** and the **intermediate representation can be optimized**, and then be executed to an back end processor. In addition, at the beginning of the current invention spec, 2nd paragraph, the inventor also mentioned that "**Intermediate representation** is a term

	widely used in the computer industry to refer to terms of abstract computer language in which a program may be expressed, but which is not specific to, and is not intended to be directly executed on, any particular processor...". Since the concept has been 'widely used' therefore it's not an invention.
14. The method of claim 1, comprising optimising the program code by optimizing the generated intermediate representation.	For the feature of claim 1 see Aho. On Aho page 463, 3rd paragraph, "A machine-independent **code optimizer** can be applied to the **intermediate representation**."
15. The method of claim 14, wherein said optimizing step is used to optimise the program code written for execution by a processor a first type so that the program code may be executed more efficiently by that processor.	For the feature of claim 14 see rejection of claim 14. Since the intermediate representation, which was generated from the program code that was written for execution by a processor, can be optimized, therefore the program code may be executed more efficiently by that

processor.

16. A method for generating an intermediate representation of program code written for running on programmable machine, said method comprising:

(i) generating plurality of register objects for holding variable values to be generated by the program code; and

(ii) generating plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;

said objects being organized into a branched tree-like network having all register objects at the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.

For the first paragraph and item (i) and (ii) see rejection on claim 1. For the last paragraph, in Aho's book, page 347, 3rd paragraph, "A convenient way to represent a type expressions is to use a graph. Using the syntax-directed approach of Section 5.2, we can construct a tree or a dag (Directed Acyclic Graph) for a type expression, with interior nodes for type constructors and leaves for basic types, type names and type variables." Where the type constructors holds the relationships in between the fixed values and the variable values. The **lowest basic root** is the leave level with the tree structure or the **tree-trunk level** of the network, and no leave would feed into any other leave (see Fig. 5.11 on Aho page 291), same as no register object feeding into any other register object.

17. A system for generating an intermediate representation of program code, comprising:

means for generating a plurality of register objects representing abstract registers, a single register object representing a respective abstract register; and

means for generating expression objects each representing a different element of said program code as that element arises the program code, each expression object being referenced by a register object to which it relates either directly, or indirectly via references from other expression objects.

See rejection of claim 1.

18. A system for generating an intermediate representation of program code written for running on a programmable machine, the system

See rejections of claim 16.

comprising:

means for generating a plurality of register objects for holding variable values to be generated by the program code; and

means for generating a plurality of expression objects representing fixed values and/or relationships between said fixed values and said variable values according to said program code;

wherein said objects are organised into a branched tree-like network having register objects the lowest basic root or tree-trunk level of the network with no register object feeding into any other register object.

Claim Rejections - 35 USC § 103

12. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

13. Claims 4, 8-11,13 are rejected under 35 U.S.C. 103(a) as being unpatentable over Aho, Alfred et al. (hereinafter "Aho") "Compilers, principles, techniques, and tools" book as applied to claims above, and further in view of U.S. Patent No. 6,463,582 by Lethin.

CLAIMS	Aho / Lethin
4. A method according to claim 1, wherein each of said steps are performed sequentially for basic blocks of said program code having only one effective entry point instruction and one effective exit point instruction.	For the feature of claim 1 see Aho. On Aho, page 528, under 'Basic Blocks' section, "A *basic block* is a **sequence of consecutive statements** in which flow of control enters at the beginning and leaves at the end without halt or possibility of branching except at the end." Aho teaches all aspects of the applicant's claims but it does not specifically mention that the basic blocks of said program code having only one effective entry point instruction and one effective exit point instruction. However, Lethin shows a basic block has only one effective entry point instruction and one effective exit

point instruction in an analogous art for the purpose of sending blocks of the original code to the compiler. In Lethin, column 23, lines 55-61, "**basic block** is a sequence of instructions where control can **only enter at the first instruction** and **can only exit at the last instruction**. This means that only the first instruction can be the target of a branch and only the last instruction can be a branch instruction. It also means that if the first instruction of the block is executed then all of the instructions will be executed."

It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the **basic blocks** of Aho with the one entry point/exit point further taught by Lethin, for the purpose of increasing efficiency to compile a heavily used portions of the currently executing program (See Lethin,

column 23, line 11-13).

8. A method according to claim 1, wherein if a said register object or a said expression object becomes redundant or unnecessary it is eliminated.

For the feature of claim 1 see Aho. Aho teaches all aspects of the applicant's claims but it does not specifically mention eliminating register object after it's no longer needed. However, Lethin shows a segment deletion method in an analogous art for the purpose of reusing computer memories. In Lethin column 43, lines 40-45, "Every segment has an **entry point counter** in it. When an entry point is deleted, compiler 104 decreases the entry point counter for the segment that contains it. When the **entry point counter** of a segment reaches **0**, no interpreter 110 are using the segment and no new interpreter 110 can jump into it. Compiler 104 deletes the segment and frees its memory for other segments to use." It would have been obvious to a person of ordinary skill in the art at the

	time of the invention was made to supplement the register allocation of Aho with the register deletion further taught by Lethin, for the purpose of increasing the efficiency of the computer memory usage (Lethin column 43, lines 28-29).
9. A method according to claim 8, wherein a redundant or unnecessary said register objéct or said expression object is identified by maintaining an ongoing count of references being made to that object as a network of register and expression objects is constructed.	See rejection of claim 8.
10. The method according to claim 9, wherein for each expression object count is maintained of the number of references to that expression object from other expression objects or from register objects, the count associated with particular expression object being adjusted	See rejection of claim 8.

each time a reference to that expression object is made or removed.

11. A method according to claim 10, wherein an expression object and all references from that expression object are eliminated when said count for that expression object is zero.	See rejection of claim 8.
13. The method claim wherein said translating step is performed dynamically as the program code is run.	On Aho, page 347, under 'Static and Dynamic Checking of Types' section, "Checking done by a compiler is said to be static, while checking done when the target program runs is termed **dynamic**." Aho teaches all aspects of the applicant's claims but it does not specifically mention that the translation step is performed as the program code is run. However, Lethin shows the concept of translating step is performed dynamically as the program code is run. In Lethin's abstract, line 1, "An optimizing object code translation system

and method perform **dynamic** compilation and translation of a target object code on a source operating system while performing optimization. **Compilation and optimization of the target code is dynamically executed in real time**." It would have been obvious to a person of ordinary skill in the art at the time of the invention was made to supplement the dynamic checking of Aho with the performing translation dynamically as the program code is run further taught by Lethin, for the purpose of improve the emulation and interpretation ability (see Lethin abstract line 6-7).

Conclusion

14. The following summarizes the status of all the claims:

102 (b) rejections: 1-3, 5-7, 12, 14, and 16-18

103 rejections: 4, 8-11, 13

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chih-Ching Chow whose telephone number is 703-305-7205. The examiner can normally be reached on 7:00am - 3:30pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q Dam can be reached on 703-305-4552. The fax phone number for the organization where this application or proceeding is assigned is 703-308-3988.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Chih-Ching Chow
Patent Examiner
Art Unit 2122

CC

JOHN CHAVIS
PATENT EXAMINER
ART UNIT 2124